# Exercise 4: Functions

Scenario 1: Calculate the age of customers for eligibility checks.

* Question: Write a function CalculateAge that takes a customer's date of birth as input and returns their age in years.

# Answer:

CREATE OR REPLACE FUNCTION CalculateAge(

p\_dob IN DATE

) *RETURN* NUMBER IS

v\_age NUMBER; BEGIN

*-- Calculate the age based on the date of birth*

v\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, p\_dob) / 12);

*RETURN* v\_age; END CalculateAge;

/

Scenario 2: The bank needs to compute the monthly installment for a loan.

* Question: Write a function CalculateMonthlyInstallment that takes the loan amount, interest rate, and loan duration in years as input and returns the monthly installment amount.

# Answer:

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment( p\_loan\_amount IN NUMBER,

p\_interest\_rate IN NUMBER, *-- Annual interest rate (e.g., 5 for 5%)*

p\_duration\_years IN NUMBER

) *RETURN* NUMBER IS

v\_monthly\_installment NUMBER; v\_monthly\_rate NUMBER; v\_num\_payments NUMBER;

## BEGIN

*-- Convert annual interest rate to monthly and calculate number of payments*

v\_monthly\_rate := p\_interest\_rate / 100 / 12; v\_num\_payments := p\_duration\_years \* 12;

*-- Calculate monthly installment using the formula*

v\_monthly\_installment := (p\_loan\_amount \* v\_monthly\_rate) /

(1 - POWER(1 + v\_monthly\_rate, -v\_num\_payments));

*RETURN* v\_monthly\_installment; END CalculateMonthlyInstallment;

/

Scenario 3: Check if a customer has sufficient balance before making a transaction.

* Question: Write a function HasSufficientBalance that takes an account ID and an amount as input and returns a boolean indicating whether the account has at least the specified amount.

# Answer:

CREATE OR REPLACE FUNCTION HasSufficientBalance( p\_account\_id IN NUMBER,

p\_amount IN NUMBER

) *RETURN* BOOLEAN IS

v\_balance NUMBER; BEGIN

*-- Fetch the balance of the account* SELECT Balance INTO v\_balance FROM Accounts

WHERE AccountID = p\_account\_id;

*-- Return true if balance is sufficient, false otherwise RETURN* v\_balance >= p\_amount;

## EXCEPTION

*WHEN* NO\_DATA\_FOUND *THEN*

*RETURN* FALSE; *-- Account not found, consider insufficient balance WHEN* OTHERS *THEN*

*RETURN* FALSE; *-- Handle any other exceptions and consider insufficient balance*

END HasSufficientBalance;

/